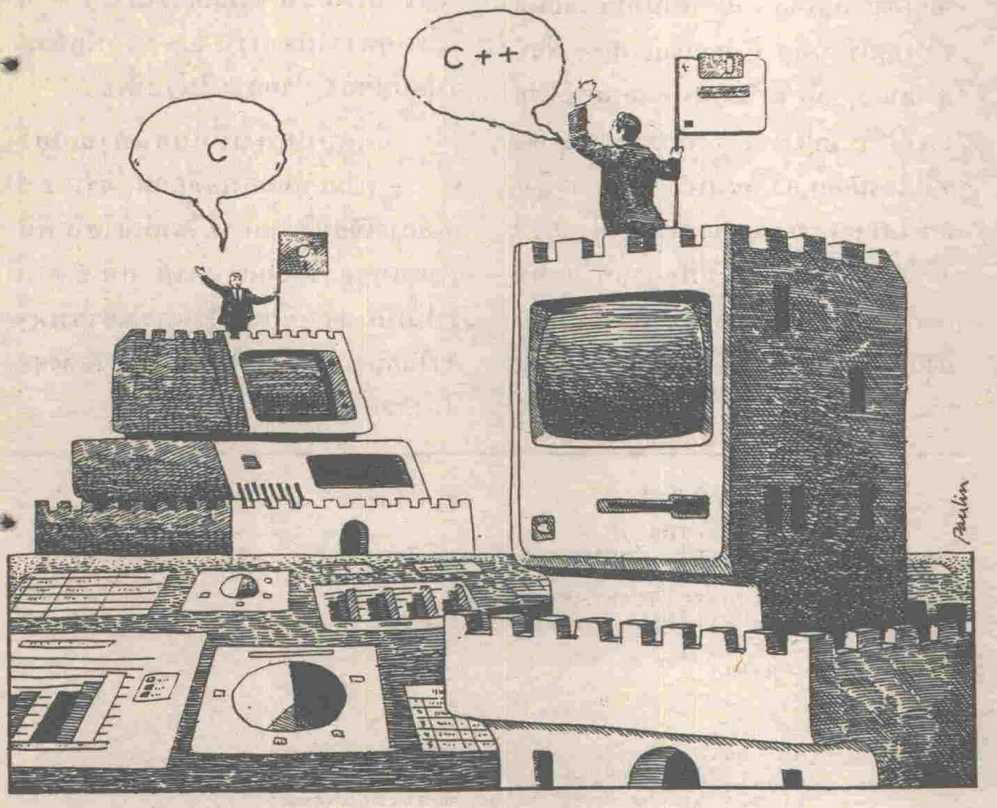
บุญเลิศ เอี่ยมทัศนา

แนะนำ เทอร์โบ C++



เรื่องราวของเทอร์โบ C++
ความสัมพันธ์และคุณสมบัติต่าง ๆ
ที่เป็นเอกลักษณ์เฉพาะตัว

ผมได้ยินเรื่องราวของ C++ มานาน แต่ไม่ค่อยจะสนใจ เพราะยังไม่เห็นความสำคัญ พอได้ทราบถึงการเขียนโปรแกรมในแบบโอโอพีจากภาษาสมอลล์ทอล์ค จึงขวนขวายหาคอมพายเลอร์ภาษาซีที่เป็น C++ จนได้ของ Zortech มา แต่ผมก็คงเช่นเดียวกับอีกหลายท่านที่คุ้นเคยกับความสะดวกสบายกับซอฟต์แวร์ของบอร์แลนด์ จึงคงเพียงแต่ศึกษา C++ กับคู่มือของ Zortech เท่านั้น พอทราบว่าบอร์แลนด์ออกเทอร์โบ C++ ก็รีบไปจัดหามา แต่ก็ต้องเก็บไว้ก่อนเพราะมีกิจอันจำเป็นทำให้ต้องห่างเครื่องอีก ขณะนี้มีโอกาสแล้วก็ขอนำมาเล่าสู่กันฟังครับ

สำหรับภาษาซี บอร์แลนด์ได้ออกโปรแกรมตัวแปรภาษามาแล้ว 3 รุ่นคือ เทอร์โบซีรุ่น 1 รุ่น 1.5 และรุ่น 2.0 (คงมีเพียงเท่านี้) แล้วมาออกรุ่นใหม่นี้โดยได้เปลี่ยนชื่อเป็นเทอร์โบ C++ แถมนับเป็นรุ่น 1 ใหม่ แต่ผมไม่คิดว่าบอร์แลนด์จะออกเทอร์โบซีธรรมดารุ่นต่อไปอีก เพราะเทอร์โบ C++ ก็คือซีที่ปรับปรุงให้เขียนโปรแกรมในแบบโอโอพีได้ด้วยเท่านั้น

โดยทั่วไป

บอร์แลนด์ได้ปรับปรุงเทอร์โบ-C++ จากเทอร์โบซีรุ่น 2 มากพอสมควร และเป็นการปรับปรุงทั้งทางด้านภาษาและสิ่งอำนวยความสะดวก ซึ่งผมจะแยกกล่าวเป็นเรื่อง ๆ ไป โดยขอนำเรื่องราวของเทอร์โบ C++ โดยทั่วไปมากล่าวถึงก่อน

เมื่อจัดหา

เมื่อคุณคิดจัดหาชุดของเทอร์โบ-C++ สิ่งที่จะได้มาคือดิสเกตต์ 8 แผ่น

กับหนังสือคู่มืออีก 4 เล่มคือ

- Getting Started
- User's Guide
- Programmer's Guide
- Reference Guide

หนังสือเล่มแรกที่ควรจะอ่านคือ Getting Started และปฏิบัติตามคำแนะนำในหนังสือนี้ ซึ่งผมเห็นว่าเขียนไว้ดี แม้แต่ผู้ไม่รู้ภาษาซีมาก่อนก็อาจทำความเข้าใจได้

ความต้องการ

ในการใช้งานเทอร์โบ C++ นี้ ความต้องการขั้นต่ำสุดสำหรับเครื่องไอบีเอ็มพีซีและเทียบเท่าคือ รุ่นเอกซ์ทีขึ้นไป หน่วยความจำหลักไม่น้อยกว่า 640 กิโลไบต์ ฮาร์ดดิสก์ 1 ตัวและฟลอปปี้ดิสก์ 1 ตัว (นั่นคือถ้าเครื่องของคุณไม่มีฮาร์ดดิสก์ก็หมดสิทธิ์ที่จะใช้เทอร์โบ C++) และใช้กับระบบจัดการดอส 2.0 ขึ้นไป สำหรับจอภาพใช้ได้กับจอภาพทุกแบบ และถ้ามีเมาส์จะช่วยให้ใช้งานเทอร์โบ C++ ได้สะดวกยิ่งขึ้น

การติดตั้ง

ความจริงแล้วเทอร์โบ C++ ได้รับการติดตั้งเสร็จเรียบร้อยมาแล้ว การรันโปรแกรม INSTALL จึงเป็นเพียงแต่ unpact โปรแกรมที่อัดแน่นกันมาในดิสก์ทั้ง 8 แผ่นลงฮาร์ดดิสก์เท่านั้น

หากคุณไม่พอใจสามารถจะติดตั้งใหม่ได้ด้วยการรันโปรแกรม TCINST และถ้าป้อนคำว่า TCCONFIG.TC ต่อท้ายจะเป็นการกำหนดการเปลี่ยนแปลงในไฟล์นี้ ซึ่งส่วนใหญ่จะเกี่ยวกับจอภาพ เช่น การกำหนดค่าสีต่าง ๆ แต่ถ้าไม่ป้อนชื่อไฟล์ดังกล่าวจะเป็นการเปลี่ยนแปลงในไฟล์ TC.EXE ซึ่งจะให้ดำเนินการเกี่ยวกับไดเรคทอรีด้วย

ใช้งาน

ในชุดของเทอร์โบ C++ มีวิธีคอมพายเลอร์อยู่ 2 แบบตามที่บอร์แลนด์กำหนดเรียกคือ Command-line ซึ่งมีลักษณะเหมือนกับคอมพายเลอร์โดยทั่วไปกับ Integrated environment ซึ่งรวมเอาเอดิเตอร์ คอมพายเลอร์และลิงเกอร์ (Linker) รวมทั้งการรันโปรแกรมไว้ด้วยกันตามแบบฉบับของบอร์แลนด์แต่เริ่มแรกมา ซึ่งในบทความนี้จะหมายถึงแบบหลังเพราะมีเรื่องราวมากกว่ากัน

และในทั้ง 2 แบบนี้สามารถเลือกการคอมพายล์ได้เป็น 2 แบบคือ ตามมาตรฐานภาษาซีของสหรัฐฯ คือ ANSI C (ANSI:American National Standard Institute) ที่บอร์แลนด์นำฉบับร่างมาใช้กับเทอร์โบซีเรื่อยมาตั้งแต่รุ่น 1.0 จนถึงรุ่น 2.0 ซึ่งในเทอร์โบ C++ ก็ไม่แตกต่างไปจากรุ่น 2.0 มากนักตามรูปแบบของโปรแกรมในรูปที่ 1

อีกแบบหนึ่งเป็นคอมพายเลอร์ C++ ตามแบบของเอทีแอนด์ทีรุ่น 2.0 เพื่อการเขียนโปรแกรมในแบบโอโอพีมีรูปแบบของโปรแกรมตามรูปที่ 2 ตามรูปนี้เป็นการเขียนโปรแกรมธรรมดายังไม่เกี่ยวกับโอโอพีมีรายละเอียดดังจะกล่าวต่อไป

```
#include <stdio.h>
#define SI 100;
float data[100];
char title[] = "First program in C++";

float square(float *val, int n );
void inVal(float *score, int n);

int main()
{


  int numVal;
  printf("%s",title);
  printf("\nHow many value must squared: ");
  scanf("%d",&numVal);
  inVal(data,numVal);
  printf("\nThe sum of the square of values = %f",
        square(data,numVal));
  return 0;
}

float square(float val[], int n)
{
  int i;

  float sum = 0.0;
  for ( i = 0; i < n; i++)
    sum += val[i] * val[i];
  return sum;
}

void  inVal(float score[], int n)
{
  int i;
  for (i = 0; i < n; i++)
  {  printf("\nEnter score %d: ",i + 1);
     scanf("%f",&score[i]);   }
}
```

รูปที่ 1 รูปแบบของโปรแกรมในแบบซี

```
// Demo program for C++
/* Old stile comment  */
#include <iostream.h>

const int SI = 100;
float data[SI];
char title[] = "First program in C++";

float square(float *val, int n = SI);

int main()
{
  void inVal(float *score, int n);

  int numVal;

  cout << title << '\n';
  cout << "How many value must squared: ";
  cin >> numVal;
  inVal(data,numVal);  cout << '\n';
  cout << "The sum of the square of values = " << square(data);
  return 0;
}

float square(float val[], int n)
{
  float sum = 0.0;
  for (int i = 0; i < n; i++)
    sum += val[i] * val[i];
  return sum;
}

void  inVal(float score[], int n)
{
  for (int i = 0; i < n; i++)
  {  cout << "\nEnter score " << i + 1 << ": ";
     cin >> score[i];  }
}
```

รูปที่ 2 รูปแบบโปรแกรมของ C++

การเลือกว่าจะใช้คอมพายเลอร์เป็นซีธรรมดาหรือ C++ สามารถดำเนินการได้ด้วยการเลือกออปชัน แต่วิธีที่ง่ายกว่าคือระบุด้วยการกำหนดนามสกุลของซอร์สไฟล์เป็น .C สำหรับซีธรรมดาหรือ .CPP สำหรับ C++

นอกจากนี้ในการคอมพายล์ยังให้สามารถเลือกได้อีกว่าจะใช้คำสงวน (Reserved word) ของ ANSI C หรือ C++ หรือ UNIX V หรือของซีดั้งเดิมที่เดนนิส ริตชี (Dennis Ritchie) กำหนดไว้ ก็คงสำหรับโปรแกรมที่เขียนไว้เดิมครับ สำหรับโปรแกรมที่จะเขียนใหม่คงจะเลือกระหว่าง ANSI C กับ C++ เท่านั้น

ความเป็นมาของ C++

แรกสุด ผมขอนำความเป็นมาของ C++ มากล่าวถึงก่อนคือ Dr. Bjarne Stroustrup แห่งห้องปฏิบัติการทดลอง Bell ได้พัฒนา C++ ต่อจากภาษาซีในปี พ.ศ.2523 เพื่อใช้ในงานสวิตชิ่งที่ซับซ้อน โดยใช้ชื่อว่า C with Classes ต่อมาได้เปลี่ยนชื่อไปเป็น C++ ตามความหมายของเครื่องหมาย ++ ในภาษาซี คือซีเพิ่มหนึ่ง

ที่แปลกคือในหนังสือ "The C++ Programming Language" ของ Dr. Stroustrup ไม่ได้พูดถึงการเขียนโปรแกรมในแบบโอโอพี และไม่ได้เอ่ยถึงภาษาสมอลล์ทอล์คเลย แต่หนังสือที่เขียนเกี่ยวกับ C++ รวมทั้งในเทอร์โบ C++ จะกล่าวถึง

ผู้พัฒนา C++ กล่าวแต่เพียงว่าได้นำแนวความคิดในเรื่องสกุล การสืบสกุล และเวอร์ชวลฟังก์ชันมาจากภาษา Simula67 กับได้นำแนวความคิดในเรื่อง overloading operators กับการที่จะให้กำหนดตัวแปร ค่าคงตัว และแบบข้อมูลในตัวโปรแกรมคือแทนที่ถ้อยแถลงได้มาจากภาษาอัลกอล 68 แล้วนำมาพัฒนาภาษาซีเป็น C++

ส่วนการที่ดำเนินการโดยพัฒนาต่อจากภาษาซีเดิมนั้นได้ให้เหตุผลว่าภาษาทุกภาษาย่อมมีข้อบกพร่อง ภาษาซีก็มีข้อบกพร่อง แต่เป็นข้อบกพร่องที่ทราบอยู่แล้ว หากสร้างภาษาใหม่ก็จะมีข้อบกพร่องใหม่อีก

สำหรับ C++ ของเอทีแอนด์ทีดังกล่าวมานี้มีอยู่หลายรุ่น ในรุ่นที่ 1.x ที่บอร์แลนด์นำมาใช้เป็นเทอร์โบ C++ คือรุ่น 2.0

เทอร์โบ C++ ในฐานะ C

จากที่กล่าวว่า C++ พัฒนาต่อจากภาษาซี ดังนั้นซีจึงควรเป็นซับเซตของ C++ นั่นคือเราสามารถที่จะใช้คอมพายเลอร์ของ C++ คอมพายล์ซอร์สโปรแกรมที่เขียนตามนิยามของซีได้ แต่เรื่องจริงไม่เป็นเช่นนี้ทั้งหมด เพราะใน C++ ได้ตัดของเดิมบางเรื่องออกไป ในเทอร์โบ C++ จึงต้องมีคอมพายเลอร์ทั้งสำหรับ ANSI C และสำหรับ C++ ดังกล่าวแล้ว

แต่ C++ ก็คือซี (ไม่ว่าจะเป็นเทอร์โบ C++ หรือของ Zortech และผมขอใช้คำว่า C++ แทนเทอร์โบ C++ เมื่อเกี่ยวกับเรื่อง C++ โดยตรง

ทั้งนี้เพื่อไม่ให้เข้าใจว่าเป็นเรื่องเฉพาะของบอร์แลนด์) ข้อแตกต่างกันเทียบได้กับเทอร์โบปาสคาลที่แตกต่างกับปาสคาลมาตรฐาน หากนำเรื่องของโอโอพีมารวมด้วยก็จะเทียบได้กับเทอร์โบปาสคาล 5.5 ต่างกับปาสคาลมาตรฐาน

ในส่วนที่ไม่เกี่ยวข้องกับโอโอพี เรื่องหลัก ๆ ที่พัฒนาบางเรื่องผมเห็นว่าคล้ายกับภาษาปาสคาลมาก กล่าวคือ (ขอให้ดูในรูปที่ 2 ประกอบกับรูปที่ 1)

- เพิ่มเครื่องหมาย // เพื่อแสดงคำอธิบาย (comment) ไปจนสุดบรรทัด

- ให้กำหนดค่าคงตัวด้วยคำว่า const แทนที่จะใช้ # define

- ให้กำหนดส่วนหัวของฟังก์ชัน (Function prototype) ไว้ก่อนที่จะใช้งาน ตามรูปที่ 2 คือ square กับ inVal (เทียบได้กับ forward reference ในเทอร์โบปาสคาล) และจากที่กำหนดเช่นนี้จะทำให้ square เป็นที่รู้จักทั่วไปในทุกฟังก์ชันที่ตามมา ส่วน inVal จะรู้จักเฉพาะใน main เท่านั้น

- ในส่วนหัวของฟังก์ชันที่กำหนดไว้ล่วงหน้านี้สามารถจะให้ค่าแก่ตัวแปรตัวท้ายได้ด้วย ดังที่ให้แก่ n ใน square ส่วนเมื่อใช้งานอาจผ่านค่าใหม่ไปแทนได้

- ได้เปลี่ยนแบบของการผ่านค่าในแบบอ้างถึงคือ pass by reference ใหม่ ดังแสดงในฟังก์ชัน swap รูปที่ 3 คือ ให้ใช้เครื่องหมาย ampersand นำหน้าตัวแปรที่ต้องการให้ผ่านค่าในแบบอ้างถึง และเมื่อเรียกใช้ฟังก์ชันดังในโปรแกรมหลักให้อ้างถึงตัวแปรโดยตรง (คือเหมือนกับในภาษาปาสคาล) ส่วนการผ่านค่าในแบบเดิมคงใช้ได้เช่นกัน ดังที่กำหนดให้ swap1

- ให้กำหนดตัวแปรในเงื่อนไขได้ ดังที่กำหนดตัวแปร i ใน for ของโปรแกรมย่อยในรูปที่ 2

- เมื่อกำหนดแบบข้อมูลแล้ว ให้นำชื่อไปใช้เป็นแบบข้อมูลใหม่เพื่อกำหนดตัวแปรได้โดยตรง (เช่นเดียวกับในภาษาปาสคาล) ดังตัวอย่าง pointT ในรูปที่ 3

- การจองและการยกเลิกหน่วยความจำในฮีพให้ใช้คำว่า new กับ delete ซึ่งจะทำงานได้ดีกว่า allocmem และ freemem

- เรื่องสุดท้ายแต่เป็นเรื่องสำคัญไม่น้อยที่ผมขอนำมากล่าวถึงคือ เอาต์พุตและอินพุต ซึ่งเดิมส่วนมากจะใช้ printf และ scanf ใน stdio.h ซึ่งมีความยุ่งยากในการใช้งานพอสมควร ให้เปลี่ยนเป็นใช้ cout และ cin แทน ดังแสดงในโปรแกรมตัวอย่างที่ผ่านมา ที่สำคัญคือ ทั้งสองคำนี้ไม่ใช่คำที่อยู่ในไลบรารี่ของ C++ หากเป็นคำในสกุล stream ที่แยกต่างหากและมีลูกเล่นในการใช้งานมากพอสมควร ขอให้ดูใน Programmer's Guide บทที่ 3 C++ stream

เรื่องอื่น ๆ ยังมีอีกครับ แต่ผมเห็นว่าเท่านี้คงจะพอมองเห็นข้อแตกต่างของ C++ กับซีในเรื่องทั่ว ๆ ไปที่ยังไม่เกี่ยวกับโอโอพี

เทอร์โบ C++ กับโอโอพี

ดังได้กล่าวแล้วว่าผู้ที่พัฒนาภาษาซีเป็น C++ ไม่ได้เอ่ยถึงโอโอพีเลย แต่ก็เป็นที่ยอมรับกันโดยทั่วไป (ในต่างประเทศ) แล้วว่าสามารถใช้ C++ (ในที่นี้หมายถึงเทอร์โบ C++) เขียน

```
// Demo type name
#include <iostream.h>

struct  pointT { int xx,yy; };
pointT pointa,pointb;

int  main()
{
  void  swap(int &frst,int &scnd);
  void  swap1(int *frst,int *scnd);
  int aa = 5, bb = 10;
  swap1(&aa,&bb);
  cout << '\n' << aa <<' ' << bb;

  cout << '\n';
  pointa.xx = 20;  pointa.yy = 5;
  pointb.xx = 40;  pointb.yy = 10;
  cout << "Point A = (" << pointa.xx << ','
                        << pointa.yy << ')'<< '\n';
  swap(pointa.xx,pointa.yy);
  cout << "After swap = (" << pointa.xx << ','
                           << pointa.yy << ')'<< '\n';
  return 0;
}

void  swap(int &frst,int &scnd)
{
  int  tem;
  tem = frst;  frst = scnd;  scnd = tem;
}

void  swap1(int *frst,int *scnd)
{
  int  tem;
  tem = *frst;  *frst = *scnd;  *scnd = tem;
}
```

รูปที่ 3 แสดงการผ่านค่าในแบบ pass by reference สองแบบ ของ swap1 คือแบบเดิม ส่วน swap คือของ C++

โปรแกรมในแบบโอโอพีได้เป็นอย่างดี ทั้งนี้โดยมองดูจากคุณลักษณะสำคัญที่มีอยู่ในภาษาสมอลล์ทอล์ค ซึ่งถือว่าเป็น OOL (Object Oriented Language) ที่แท้จริง กล่าวคือ

● **ความมุ่งหมาย**

เรื่องนี้ไม่อยู่ในคุณลักษณะสำคัญโดยตรง แต่ผมเห็นว่ามีความสำคัญในการทำความเข้าใจกับ OOL ต่อไป จึงขอนำมากล่าวก่อนคือ "ความมุ่งหมายหลักในการเขียนโปรแกรมในแบบโอโอพีก็เพื่อที่จะพัฒนาโปรแกรมให้เร็วที่สุดโดยไม่ค่อยคำนึงถึงขนาดของโปรแกรมและความเร็วในการรัน" (การพัฒนาโปรแกรมหมายรวมถึงงานเขียน ป้อนคอมพายล์ ลิงก์ รันทดสอบ แก้ที่ผิดและปรับปรุงจนโปรแกรมใช้งานได้ดี)

● **เป็นเอกเทศ (Encapsulation)**

คุณลักษณะสำคัญประการแรกที่ OOL ต้องมีคือความเป็นเอกเทศหมายถึงว่า เราสามารถเขียนโปรแกรมให้เก็บข้อมูลและโปรแกรมย่อยที่จะดำเนินการกับข้อมูลเหล่านี้ไว้ในที่เดียวกันได้และอยู่ในปกป้องคือ สำหรับข้อมูลแล้วไม่ให้ผู้หนึ่งผู้ใดเข้าถึงได้เลย จะต้องดำเนินการผ่านโปรแกรมย่อยที่กำหนดให้ ส่วนโปรแกรมย่อยซึ่งตามโอโอพีจะเรียกว่ากิจกรรม (method) สามารถกำหนดได้ว่ากิจกรรมใดที่จะใช้เป็นการภายในหรือกิจกรรมใดให้เรียกใช้ได้

ซึ่ง C++ (หมายถึงเทอร์โบ C++ ด้วยตามที่กล่าวแล้ว) ดำเนินการโดยกำหนดแบบข้อมูลแบบ class (มีลักษณะคล้ายเรคอร์ดในภาษาปาสคาลหรือ struct ในภาษาซีและสามารถใช้ struct ได้เช่นเดียวกับ class แต่แตกต่างกันในรายละเอียด) เพื่อการกำหนดสกุล (Class)

ตามรูปที่ 4 เป็นการกำหนดสกุล location มีตัวแปร x และ y เพื่อเก็บข้อมูลที่เป็นโคออร์ดิเนตของจุด และมีกิจกรรม setpstn สำหรับให้ค่าแก่ตัวแปรทั้งสองนี้ และกิจกรรม retx กับ rety เพื่อการอ่านค่าตัวแปรทั้งสองนี้

ถึงตรงนี้ผมคงต้องกล่าวถึงเรื่องอื่นก่อนครับ คือจุดประสงค์ของผมต้องการแสดงว่า C++ สามารถใช้เขียนโปรแกรมที่เป็นเอกเทศได้อย่างไร แต่มีเรื่องประกอบที่จำเป็นต้องกล่าวถึงก่อนคือ

- ทั้งตัวแปรและโปรแกรมย่อย (กิจกรรม) ดังที่กล่าวมานี้ยังไม่มีตัวตนจริง แต่เป็นการกำหนดรูปแบบไว้เท่านั้น เรียกว่าเป็น abstract data type

- สำหรับตัวแปรจะมีตัวตนจริงคือมีการจองเนื้อที่ในหน่วยความจำ เมื่อกำหนดตัวแปรของสกุล (instance) ซึ่งผมขอใช้คำในภาษาไทยว่า "นิมิต" เพื่อไม่ให้สับสนกับตัวแปรอื่นเพราะมีลักษณะเฉพาะดังที่กำหนดให้แก่ point1 และ point2 หมายถึงจุดหนึ่ง ๆ บนจอภาพ จะกำหนดกี่ตัวก็ได้ ที่กำหนดไว้ 2 ตัว เพื่อแสดงให้เห็นว่ากำหนดได้หลายตัว และในแต่ละจุดนี้จะมี x กับ y เป็นของตนเอง เช่นเดียวกับข้อมูลแบบเรคอร์ด คือ struct โดยทั่วไป

- ส่วนโปรแกรมย่อย หากเป็นภาษาสมอลล์ทอล์คจะเก็บอยู่ในส่วนที่เก็บโปรแกรมย่อย ยังไม่ได้นำมาเชื่อมโยงกับสกุล แต่จะนำมาเชื่อมโยงในขณะรัน (runtime) เรียกว่า Late binding ซึ่งมีทั้งข้อดีและข้อเสีย ข้อดีคือสามารถทำให้เขียนโปรแกรมในแบบหลายการปฏิบัติ (Polymorphism) ได้ ซึ่งเป็นลักษณะเฉพาะที่สำคัญของ OOL ส่วนข้อเสียคือใช้เวลาในการรันมากขึ้น

- แต่สำหรับ C++ ตามโปรแกรมนี้จะเชื่อมโยงโปรแกรมย่อยเข้ากับสกุลในขณะคอมพายล์ จะให้เป็น late

```
// Example 4
// Demo Encapsulation

#include <iostream.h>
#include <graphics.h>

class  location {
  int  x,y;

  public:
    void setpstn(int newx, int newy) {x = newx;  y = newy;}
    int  retx()  { return x; }
    int  rety()  { return y; }
};

location  point1,point2;

int  main()
{
  int  curx,cury;

  cout << "Set coordinate x and y: ";
  cin  >> curx >> cury;
  point1.setpstn(curx,cury);
  cout << "the coordinat of point 1 is ";
  cout << point1.retx() << ' ' << point1.rety() << '\n';

  return 0;
}
```

รูปที่ 4 แสดงการกำหนดสกุล location

binding เฉพาะในกรณีที่ต้องการให้มีหลายการปฏิบัติ ดังจะกล่าวต่อไป

จากการกำหนดนิมิตนี่แหละครับที่จะแสดงความหมายของคำว่าเป็นเอกเทศได้ กล่าวคือ ถ้าเป็นเรคอร์ดตามปกติเราสามารถจะใช้ถ้อยแถลงต่อไปนี้ได้คือ

```
point1.X = 5;
```

ซึ่งเป็นการเข้าถึงข้อมูลโดยตรง แต่ใน C++ สำหรับแบบข้อมูล class ทำไม่ได้ ส่วนกิจกรรมก็เช่นกัน แต่ตามตัวอย่างนี้ให้เข้าถึงได้ คือสามารถใช้ถ้อยแถลงต่อไปนี้ได้

```
point1.set (5,10);
curx = point1.retx
```

เป็นการเข้าถึงข้อมูลตามที่กำหนดให้ทำได้ สำหรับกิจกรรมที่กำหนดให้เข้าถึงไม่ได้จะเห็นในตัวอย่างต่อไป

นี่คือความเป็นเอกเทศในนัยของโอโอพี แต่ทั้งนี้ขอให้รวมถึงการที่สกุลแต่ละสกุลมีความพร้อมในตัวไม่ต้องพึ่งพาใครอื่นอีกไว้ด้วย และผมเห็นว่าคำว่า "เอกเทศ" ให้ความหมายของคุณลักษณะเฉพาะในเรื่องนี้ได้ชัดเจนกว่าคำว่า Encapsulation

จากที่กล่าวมานี้ผมคิดว่าคงไม่แสดงให้เห็นถึงความสำคัญของความเป็นเอกเทศได้มากนัก ก็ต้องเชื่อไว้ก่อนครับว่ามีความสำคัญยิ่งและเป็นพื้นฐานของคุณลักษณะข้ออื่น

แต่ความเป็นเอกเทศในด้านที่ให้เข้าถึงข้อมูลไม่ได้เลยนั้นก็มีข้อเสียอยู่เหมือนกัน เพราะต้องดำเนินการผ่านโปรแกรมย่อย ทำให้ใช้เวลาในการรันมากขึ้น ดังนั้นใน C++ ซึ่งไม่เคร่งในเรื่องนี้จึงมีวิธีที่จะทำให้เข้าถึงข้อมูลได้ ทำให้มีข้อกล่าวหาว่า C++ ไม่ใช่ OOL ที่แท้จริง

● สืบสกุล (Inheritance)

การสืบสกุลเป็นคุณลักษณะเฉพาะที่สำคัญของ OOL เช่นกัน หมายถึงเมื่อเรากำหนดสกุลแล้ว ยังสามารถกำหนดสกุลรองหรืออาจเรียกว่าเป็นสกุลลูกให้เป็นผู้สืบสกุลของสกุลเดิมได้ โดยสกุลลูกสามารถใช้กิจกรรมของสกุลพ่อได้ (แต่กลับกันไม่ได้)

สำหรับ C++ สามารถดำเนินการได้ดังตัวอย่างในรูปที่ 5 ซึ่งเป็นการกำหนดสกุล point ให้เป็นผู้สืบสกุลของสกุล location ในตัวอย่างนี้ได้แสดงถึงการเรียกใช้กิจกรรมในสกุลพ่อทั้งในการกำหนดฟังก์ชัน set และการเขียนโปรแกรมย่อยของ hide ก็เพื่อแสดงให้เห็นว่า C++ สามารถสืบสกุลได้เท่านั้น

ยังมีเรื่องอื่นที่จะต้องชี้แจงควบคู่กันก่อนคือ โปรแกรมตามรูปที่ 4 นั้น ผมเขียนอย่างสมบูรณ์สามารถนำไปรันได้โดยใช้ inline functions เข้าช่วย แต่ไม่ถูกวิธีการเขียนโปรแกรมในแบบโอโอพี

วิธีที่ถูกต้องขอให้ดูในรูปที่ 6 ซึ่งจะเขียนการกำหนดสกุล กรณีนี้คือ location ไว้ในไฟล์ .h เก็บไว้ ส่วนตัวโปรแกรมกิจกรรมทั้งหมด (ตามรูปแสดงไว้เพียงโปรแกรมเดียวคือ hide)

```
//  Example 5

class  point : public location {
  public:
    void setpoint(int newx, int newy) { setpstn(newx,newy); }
    void show();
    void hide();
    void moveto(int pstnx, int pstny);
};


void  point::hide(void)
{ putpixel(retx(),rety(),getbkcolor()); }
```

รูปที่ 5 แสดงการกำหนดสกุล point เป็นผู้สืบสกุลของ location ได้แสดงกิจกรรมใน point ไว้เพียงกิจกรรมเดียวเพื่อเป็นตัวอย่างเท่านั้น

```
// Example 6

// file  LOCATE.H
class  location {
  int  x,y;

  public:
    void setpstn(int newx, int newy);
    int  retx();
    int  rety();
};


// file  LOCATEF.CPP
void  location::setpstn(int newx, int newy)
{  x = newx;  y = newy; }

int   location::retx(void)
{  return x;  }

int   location::rety(void)
{  return y;  }
```

รูปที่ 6 แสดงการแยกไฟล์ของสกุล locate

จะเขียนไว้ในไฟล์ .cpp แล้วคอมพายล์เป็นไฟล์ .obj เก็บไว้ (สมมติว่าตั้งชื่อไฟล์ไว้เป็น LOCATE.H, LOCATEF.CPP จะได้ไฟล์ LOCATEF.OBJ) ก็จะกำหนดสกุล point ได้ดังรูปที่ 7 ซึ่งแสดงเพียงบางส่วนของโปรแกรมเท่านั้น ขอให้สังเกตว่าผมได้ผนวก (include) ไฟล์ locate.h และได้ให้คำแนะนำไว้ว่าให้ลิงก์กับไฟล์ locatef.obj นี่คือการเขียนโปรแกรมตามแบบโอโอพีที่ถูกต้อง แต่ตัวสกุล point เองยังเขียนรวมอยู่กับ main ถ้าจะให้ถูกต้องจริง ๆ ต้องแยกออกดังกล่าวมาแล้ว

นี่คือประโยชน์ของการสืบสกุล คือต้องนึกว่าสกุลเดิมจะอยู่ในไฟล์ .h ซึ่งเราอ่านได้ กับไฟล์ .obj ซึ่งอ่านไม่ได้ และถ้าเป็นงานที่ซับซ้อนจะไม่ทราบว่าผู้เขียนโปรแกรมเดิมได้ดำเนินการไว้อย่างไรบ้าง ซึ่งเป็นเรื่องที่ไม่จำเป็นต้องทราบ เพราะจากการที่เราสามารถสืบสกุลได้นี้ ทำให้เราสามารถเพิ่มเติมหรือดัดแปลงโปรแกรมเดิมได้โดยไม่ต้องแตะต้องกับซอร์สโค้ดเดิม ทำให้ช่วยประหยัดเวลาในการพัฒนาโปรแกรมได้มาก

ในทางตรงกันข้ามหากเราเขียนโปรแกรมด้วยภาษาซีธรรมดา เมื่อจะปรับปรุงโปรแกรมจะต้องนำซอร์สโค้ดเดิมมาศึกษาก่อนจึงจะปรับปรุงโปรแกรมได้ ทำให้เสียเวลามาก ส่วนข้อเสียของโอโอพีคือตัวโปรแกรมจะมีขนาดใหญ่กว่าและมีส่วนที่ไม่ได้ใช้ แต่บอร์แลนด์กล่าวว่าได้แก้ข้อเสียนี้โดยจะไม่นำโปรแกรมย่อยที่ไม่ได้ใช้มารวมด้วย

```
//  Example 7

#include "locate.h"
#include <graphs.h>

/*  link with  LOCATEF.OBJ  */

class  point : public location {
  public:
    void setpoint(int newx, int newy) { setpstn(newx,newy); }
    void show();
    void hide();
    void moveto(int pstnx, int pstny);
};


void  point::hide(void)
{ putpixel(retx(),rety(),getbkcolor()); }


int  main()
{

}
```

รูปที่ 7 แสดงการกำหนดสกุล point เป็นผู้สืบสกุลของ locate จากไฟล์ locate.h

● หลายการปฏิบัติ (Polymorphism)

คำว่า Polymorphism มาจากภาษากรีกหมายถึง "having many shapes" ผมจึงขอใช้คำว่าหลายการปฏิบัติ (เพราะยังหาคำที่เหมาะสมกว่าไม่ได้) ตามลักษณะเฉพาะของมัน ซึ่งต้องขออธิบายประกอบกับโปรแกรมตัวอย่างดังรูปที่ 8 ซึ่งแสดงเฉพาะการกำหนดสกุลต่าง ๆ เท่านั้น เริ่มจาก location แล้วนำไปกำหนดสกุลรอง point แล้วนำไปกำหนดสกุล circle และได้แสดงตัวอย่างใช้งานใน main ( )

จะเห็นว่าในสกุล point และ circle มีกิจกรรมชื่อเดียวกันอยู่ 2 กิจกรรมคือ hide และ show ซึ่งมีความหมายเหมือนกันคือ hide หมายถึงให้ลบภาพบนจอภาพ ส่วน show หมายถึงให้แสดงภาพบนจอภาพ แต่การปฏิบัติแตกต่างกันกล่าวคือ

- ใน point จะเป็นการดำเนินการต่อภาพเพียง 1 จุดภาพเท่านั้นคือ ลบภาพ 1 จุดภาพหรือแสดงภาพ 1 จุดภาพ

- ส่วน circle เป็นการดำเนินการต่อภาพทั้งวงกลมคือ ลบภาพ 1 วงกลมหรือแสดงภาพ 1 วงกลม

ที่เรียกว่าเป็นหลายการปฏิบัติคือในกิจกรรม jumpto จะเรียกใช้ hide และ show ซึ่งจะเห็นว่าใน circle ไม่ได้กำหนดกิจกรรม jumpto ไว้ แต่นำมาใช้ใน main ( ) คือ p.jumpto และ c.jumpto โดยที่เมื่อใช้ c.jumpto จะไปทำ jumpto ( ) ใน point แต่ทำ hide ( ) และ show ( )ใน circle

ความสามารถที่ให้เป็นหลายการปฏิบัติได้เช่นนี้ ทำให้มีความอ่อนตัวในการปรับปรุงโปรแกรมมากยิ่งขึ้น ไม่ว่าเราจะเปลี่ยนออปเจ็กต์โดยส่วนรวมดังเช่นกรณีของ point กับ circle หรือในกรณีที่ต้องการเปลี่ยนแปลงข้อมูล เช่น เดิมกำหนดแบบข้อมูลไว้เป็นอะเรย์ แต่ต่อมาต้องการเปลี่ยนแบบข้อมูลเป็นพอยน์เตอร์ไปยังส่วนอะเรย์ก็สามารถดำเนินการได้โดยไม่กระทบกับโปรแกรมที่ใช้สกุลเดิมอยู่

เรื่องนี้ C++ ดำเนินการโดยให้เรากำหนดให้ฟังก์ชันดังกล่าวมานี้เป็น

Virtual ดังที่เห็นในรูปที่ 8 และเรียกว่า virtual function ตามที่กล่าวในตอนต้นแล้วว่านำมาจากภาษา Simula67

● ซ้ำซ้อน (Overloading)

หมายถึงให้สามารถกำหนดฟังก์ชันชื่อเดียวกันซ้อนกันได้ เช่นต้องการเขียนโปรแกรมยกกำลังสามของเลขจำนวนเต็มและจำนวนจริง หากเขียนด้วยซีธรรมดาจะเขียนได้ดังรูปที่ 9 ท่อนบน แต่ถ้าเขียนด้วย C++ จะเขียนได้ดังรูปที่ 9 ท่อนล่าง (ทั้งหมดนี้แสดงเฉพาะส่วนหัว)

และไม่ใช่เฉพาะชื่อฟังก์ชันเท่านั้น สามารถกระทำกับตัวดำเนินการ (operator) ได้ด้วย เช่น + อาจไม่ใช่เป็นเพียงแต่การบวกเลขเท่านั้น แต่อาจให้รวมภาพตัวต่อสตริง หรืออื่น ๆ อีก

สิ่งอำนวยความสะดวก

ตามที่บอร์แลนด์เรียกว่า IDE (Integrated Development Environment) และที่ผมเรียกว่าเทอร์โบคือโปรแกรมที่จะให้เราป้อนคอมพายล์ ลิงก์ และรันโปรแกรมในภาษา...กรณีนี้คือภาษาซีได้โดยสะดวก

ครั้งนี้ทางบอร์แลนด์ได้ปรับปรุงเทอร์โบไปจากเดิมมากมายที่เด่น ๆ คือให้สามารถเปิดหน้าต่างหลายบานซ้อนทับกันได้ เลื่อน ขยายหรือลดขนาดหน้าต่างได้ ลอกข้อความจากหน้าต่างหนึ่งไปอีกหน้าต่างหนึ่งได้ สามารถใช้เมาส์ได้และอื่น ๆ อีก ทำให้การใช้งานหน้าต่างค่อนข้างจะวุ่นวายพอสมควร เปรียบได้กับเครื่องเสียงที่มีปุ่มให้ปรับมากมาย

C++ กับ C

เท่าที่กล่าวมานี้ก็เพื่อแสดงให้เห็นว่าเทอร์โบ C++ มีคุณลักษณะที่สำคัญ

```
// figures.h contains three classes.

class Location {
protected:
   int X;
   int Y;
public:
   Location(int newX, int newY) {X = newX; Y = newY;}
   int GetX() {return X;}
   int GetY() {return Y;}
};

class Point : public Location {
protected:
   Boolean Visible;
public:
   Point(int newX, int newY);
   virtual void Show();        // Show and Hide are virtual
   virtual void Hide();
   void jumpto(int NewX, int NewY);
};

class Circle : public Point {  // Derived from class Point and
                               // ultimately from class Location
protected:
   int Radius;
public:
   Circle(int newX, int newY, int newRadius);
   void Show();
   void Hide();
};

// ========================================

main()
{
  Point  p;
  Circle  c;

// ....................................

  p.jumpto(46,15);
  c.jumpto(25,20);

// ....................................
```

รูปที่ 8 แสดงการกำหนดกิจกรรมให้เป็นเวอร์ชวล
(หมายเหตุ : กิจกรรมที่มีชื่อซ้ำกับสกุลคือ กิจกรรมที่เป็น constructor ซึ่งไม่ได้กล่าวในบทความนี้)

```
  init  icube(int num);
  float fcube(float fnum);

  int   cube(int num);
  float cube(float fnum);
```

รูปที่ 9 แสดงการกำหนดฟังก์ชันให้ใช้ชื่อซ้ำซ้อนกันได้ใน C++

ตามที่ภาษา OOL ต้องการ ส่วนการปรับปรุงทางด้านภาษานั้นแยกเป็น 2 ตอนคือ เป็นการปรับปรุงให้ตรงตาม ANSI C ซึ่งเทอร์โบซี 2.0 ก็ใกล้เคียงอยู่แล้ว อีกตอนเป็นการปรับปรุงให้ตรงตาม C++ ของเอทีแอนด์ที รุ่น 2.0 ซึ่งแตกต่างไปจาก ANSI C

ดังนั้นหากคุณจะจัดหามาเพื่อใช้เขียนโปรแกรมภาษาซีก็ต้องคิดสักหน่อยครับ เพราะต้องใช้เนื้อที่ในดิสก์ถึง 6 เมกะไบต์ และตัว TC.EXE เองก็มีขนาดถึงประมาณ 800 กิโลไบต์ ดังนั้นหากเครื่องของคุณมีหน่วยความจำหลักเพียง 640 กิโลไบต์ ก็จะต้องติดต่อกับดิสก์ตลอดเวลา ทำให้ทำงานได้ช้ามาก

ต่ถ้าต้องการเขียนโปรแกรมในแบบ
อโอพีก็ต้องพิจารณากันต่อไป

ทอร์โบ C++ กับสมอลล์ทอล์ค

ภาษาสมอลล์ทอล์คเป็นภาษาที่ถือกันว่าเป็น OOL อย่างแท้จริง แต่ผมกลับเห็นว่าน่าจะเป็นเพียงภาษาในห้องทดลองมากกว่า แต่ทั้งนี้ผมก็รู้จักภาษาสมอลล์ทอล์คเพียงตัวเดียวคือ "สมอลล์ทอล์ค 5" ที่มีความประทับใจในทางลบเป็นอย่างมากในการทำงานด้วย โดยเฉพาะเอดิเตอร์และเวลาที่ใช้ในการรัน

สำหรับ C++ นั้น ผู้ที่นิยมสมอลล์ทอล์ค (ในต่างประเทศ) วิจารณ์ว่าไม่ใช่ OOL ที่แท้จริง ซึ่งผมมองดูแล้วน่าจะมีเพียงแห่งเดียวตรงความเป็นเอกเทศ ซึ่งถือว่าภาษาที่เป็น OOL จะต้องปกป้องข้อมูลและกิจกรรมได้ถึง 3 ระดับคือ

- โปรแกรมส่วนอื่นเข้าถึงไม่ได้
- นิมิตของสกุลนั้น ๆ เข้าถึงไม่ได้
- ผู้สืบสกุลเข้าถึงไม่ได้

ซึ่งใน C++ ก็ดำเนินการได้แต่ไม่ปิดสนิทที่เดียวนักเพราะ C++ เป็นภาษาใช้งาน จึงเห็นว่าในการเข้าถึงข้อมูลหากต้องดำเนินการผ่านโปรแกรมย่อยทุกครั้งจะเสียเวลาไปโดยใช่เหตุ จึงมีวิธีที่จะให้ผู้สืบสกุลเข้าถึงได้

อีกประเด็นหนึ่งคือใน C++ ต้องใช้ไฟล์ .h ดังนั้นเรื่องใดที่ผู้เขียนโปรแกรมต้นสกุลกำหนดไว้ในไฟล์นี้ ผู้เขียนโปรแกรมคนต่อไปสามารถเรียกมาดูและแก้ได้ ซึ่งอาจทำให้เกิดปัญหาได้ แต่เรื่องนี้ในสมอลล์ทอล์ค 5 ก็ทำได้เช่นกัน

เทอร์โบ C++ กับเทอร์โบปาสคาล 5.5

บอร์แลนด์ได้ปรับปรุงเทอร์โบปาสคาลของตนให้เขียนโปรแกรมในแบบโอโอพีได้เช่นกันในรุ่น 5.5 และออกมาก่อนหน้าเทอร์โบ C++ ซึ่งในเทอร์โบปาสคาล 5.5 เองก็กล่าวไว้ในหนังสือ "The C++ Programming Language" โดย Bjarne Stroustrup

แต่สิ่งที่ขาดอยู่คือเทอร์โบปาสคาล 5.5 ยังไม่มีขีดความสามารถในเรื่องซับซ้อน (overloading) กับไม่ปกป้องข้อมูลและกิจกรรมในสกุลบรรพบุรุษต่อนิมิตและสกุลลูก ซึ่งเรื่องนี้ผมเห็นว่าไม่เป็นข้อบกพร่องเท่าไร เพราะใน C++ ปกป้องก็จริงแต่มีทางให้เข้าถึงได้ซึ่งเท่ากับไม่ได้ปกป้องแล้ว

และในกรณีของไฟล์ .h เทอร์โบปาสคาล 5.5 ไม่ได้ให้คอมพายล์สกุลเก็บไว้ในยูนิต (unit) ดังนั้นหากไม่มีซอร์สโปรแกรมแล้ว ผู้เขียนโปรแกรมคนหลัง ๆ จะดัดแปลงแก้ไขเรื่องราวในสกุลเดิมไม่ได้เลย ซึ่งผมคิดว่าให้ความปลอดภัยมากกว่า

แต่ทั้งนี้เป็นคำพูดของผู้ที่คุ้นเคยกับภาษาปาสคาลมากกว่าภาษาซีเพราะผมไม่ค่อยชอบความหลากหลายในภาษาซี เช่น มีคำในไลบรารี่มากมายจนเลือกใช้แทบไม่ถูก มีตัวดำเนินการ (operator) มากมาย ใช้เครื่องหมายหลายตัวหลายแห่งหลายที่จนแทบจำไม่ได้ แต่ผมก็เช่นเดียวกับอีกหลายท่านครับคือเมื่อเขียนโปรแกรมในภาษาซีไปนาน ๆ ก็ไม่อยากกลับไปเขียนโปรแกรมในภาษาปาสคาล เพราะเขียนโปรแกรมได้กะทัดรัดกว่ากัน

ในสองภาษานี้จึงขึ้นอยู่กับความพอใจที่จะเลือกใช้ภาษาใด และผมคิดว่าบอร์แลนด์คงปรับปรุงให้เหมือนกับ C++ ในเทอร์โบปาสคาลรุ่น 6.0

มองไปข้างหน้า

ตามที่กล่าวว่าความมุ่งหมายหลักของการเขียนโปรแกรมในแบบโอโอพีคือให้พัฒนาโปรแกรมได้เร็ว แต่เมื่อกำหนดสกุลแล้วกลับมีความเป็นโอโอพีในอีกด้านหนึ่งคือ เป็นลักษณะของโปรแกรมที่ยังไม่สมบูรณ์ แล้วให้คนอื่นไปเขียนโปรแกรมต่อเติมต่อไป เช่น เขียนโปรแกรมสำหรับตัวหมากรุกและการเดินซึ่งต้องการความรู้ทางด้านกราฟิก แล้วให้ผู้ที่มีความรู้ทางด้านการเล่นหมากรุกนำไปเขียนโปรแกรมเกมหมากรุก

หรืออย่างโปรแกรมกึ่งสำเร็จรูป เช่น เมื่อเราจะใช้ดีเบสก็ต้องศึกษาภาษาดีเบส เมื่อจะใช้โลตัส 1-2-3 ก็ต้องศึกษาภาษานี้ จึงเป็นไปได้ไหมว่าในอนาคตอาจเขียนด้วยภาษา OOL เช่น C++ และให้ผู้ใช้เขียนโปรแกรมใช้งานด้วย C++ โดยไม่ต้องศึกษาภาษาเฉพาะเรื่องใหม่ทุกครั้งที่เปลี่ยนซอฟต์แวร์

บริษัทซอฟต์แวร์เดิมอาจไม่เห็นด้วย เพราะวิธีเดิมเป็นวิธีหนึ่งที่จะผูกพันลูกค้าของตนไว้ แต่บริษัทซอฟต์แวร์ใหม่อาจแย่งตลาดด้วยวิธีใหม่นี้ ก็ต้องดูกันต่อไปครับ